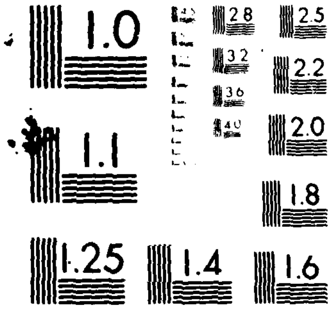
AD-A110 797   BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA   F/G 5/9
PROJECT STEAMER: VI. ADVANCED COMPUTER-AIDED INSTRUCTION IN PRO--ETC(U)
JAN 82   A STEVENS, B ROBERTS, L STEAD   N00123-81-D-0794

UNCLASSIFIED   NPRDC-TR-82-28   NL

END
DATE
FILMED
3 82
DTIC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

NPRDC TR 82-28

January 1982

PROJECT STEAMER: VI. ADVANCED COMPUTER-AIDED INSTRUCTION IN PROPULSION ENGINEERING--AN INTERIM REPORT

Albert Stevens
Bruce Roberts
Larry Stead
Kenneth Forbus
Cindy Steinberg
Brian Smith

Bolt Beranek and Newman, Inc.
Cambridge, MA 02138

Reviewed by
John D. Ford, Jr.

Released by
James F. Kelly, Jr.
Commanding Officer

Navy Personnel Research and Development Center
San Diego, California 92152

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER
NPRDC TR 82-28

2. GOVT ACCESSION NO.
AD-A110797

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
PROJECT STEAMER: VI. ADVANCED COMPUTER-AIDED INSTRUCTION IN PROPULSION ENGINEERING--AN INTERIM REPORT

5. TYPE OF REPORT & PERIOD COVERED
Interim Report
Mar 1980-Feb 1981

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

Albert Stevens
Bruce Roberts
Larry Stead
Kenneth Forbus
Cindy Steinberg
Brian Smith

8. CONTRACT OR GRANT NUMBER(s)
N00123-81-D-0794

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Bolt Beranek and Newman, Inc.
Cambridge, MA 02138

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
63720N
Z1177-PN.03

11. CONTROLLING OFFICE NAME AND ADDRESS
Navy Personnel Research and Development Center
San Diego, California 92152

12. REPORT DATE
January 1982

13. NUMBER OF PAGES
28

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Propulsion engineering
Intelligent computer-assisted instruction
Artificial intelligence
Computer graphics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

At present, propulsion engineering training is a major problem for the Navy. NAVPERSRANDCEN's goal is to build an intelligent computer-aided instruction (ICAI) training system to enhance Navy training in this area and evaluate the potential of new computer-based technology to provide a qualitatively different form of training. The STEAMER project is concerned with the design and implementation of a training system to instruct personnel in the principles of propulsion engineering. Operational personnel need to understand these principles of plant operation in order to remember the multitude of procedures required for safe plant operation and to be able to respond to a myriad of potential casualty conditions.

This report describes the current STEAMER prototype. This system consists of a graphical interface to a simulation of a 1200 psi propulsion system. This graphical interface to the mathematical model provides students with an easy and natural method of inspecting and manipulating the plant simulation. At this point in the STEAMER development, the 19E22 mathematical model has been translated into a language (Lisp) that will facilitate the addition of explanation and tutorial facilities and a color graphics interface to the model has been developed. This system currently exists on a large mainframe computer and is being moved to a portable Lisp machine for further development.

FOREWORD

This research and development was conducted in support of Navy Decision Coordinating Paper Z1177-PN (Advanced Computer-Aided Instruction), subproject Z1177-PN.03 (STEAMER: Advanced Computer-Based Training for Propulsion and Problem Solving). It was sponsored by the Chief of Naval Operations (OP-01). The main objective of the STEAMER effort is to develop and evaluate advanced knowledge-based techniques for use in low-cost portable training systems. The project is focused on propulsion engineering as a domain in which to investigate these computer-based training techniques.

This report, the sixth in a series on the STEAMER project, describes the current STEAMER prototype, including the mathematical model, graphics, explanation generation, procedures training, and minilabs, as well as the basic support software for further STEAMER development. Previous reports described an initial framework for developing techniques for automatically generating explanations of how to operate complex physical devices; a user's manual for the STEAMER interactive graphics package; a method for generating explanations using qualitative simulation; CONLAN, a constraint-based programming language well suited for describing and analyzing complex devices; and a mathematical simulation of the STEAMER propulsion plant (NPRDC TNs 81-21, 81-22, 81-25, 81-26, and 81-27, respectively). Intended users of this series of reports are system maintainers and other research personnel.

Appreciation is expressed to the staff of the Surface Warfare Officer School in Newport, Rhode Island, especially CDR Bissonnette, LCDR Ogurek, LCDR Hunt, LCDR Bowler, Senior Chief Machinist Mate Liptak, and Chief Tradevman Henley, for their participation in beneficial discussions about the nature of the propulsion engineering training problem and for specific advice on the STEAMER displays and user interface.

Dr. J. Hollan was the contracting officer's technical representative.

JAMES F. KELLY, JR. Commanding Officer	JAMES J. REGAN Technical Director

SUMMARY

Problem

Naval officers, technicians, and operators have insufficient opportunity to practice complex skills such as those involved in the operation of propulsion engineering plants. For many complex skills, increased practice is so prohibitively expensive that Navy personnel have had only a minimal level of practice. Currently, a technological opportunity exists to increase dramatically the amount of practice in propulsion plant operation and to provide a qualitatively different form of training.

Objective

The main objective of the STEAMER effort is to take advantage of this technological opportunity to meet a critical need for improvement of propulsion engineering training. An inexpensive desktop-sized computer training system will be produced that will greatly increase the amount and quality of training available to Navy personnel. The objective of this report is to describe the current prototype of STEAMER.

Approach

A training system will be developed that will allow the student to "operate" a propulsion plant and to understand its functioning. The effort will provide a form of training that allows a student to interact with a simulation of a propulsion plant by means of a graphical display, to inspect the plant at a variety of conceptual levels during its operation, and be given explanations and tutorial assistance during the interaction. All current trends indicate that computer hardware capable of supporting such a system will cost about $10,000 in 1985 and be small enough to fit on a desktop.

Results

The STEAMER system consists of a simulation of a propulsion plant that can be displayed as animated diagrams and controlled using a graphics interface. Using this interface, the student can manipulate simulated components such as valves, switches, and pumps and observe the effects on plant parameters such as changes in pressures, temperatures, and flows. The tutorial component of STEAMER presently exists as a set of prototype pieces that will be refined, tested, and integrated into an intelligent tutorial component of the system. These prototype pieces include one for generating explanations of component operations, one for teaching basic physics principles, and one for providing guidance on plant operating procedures.

Future Direction

The complete STEAMER prototype is still being developed and evaluations of its use by operational personnel are just beginning. The overall system design emphasizes flexibility. The simulation is controlled by a mathematical model of a 1078-class steamplant. Diagrams and connections to the mathematical model can be easily modified or added. Software can be easily moved to several other types of processors. Further development and evaluation are expected to continue during this year with small pieces of the complete system moved to microprocessors for use and evaluation in Navy schools.

CONTENTS

LIST OF FIGURES

INTRODUCTION

Problem

Naval officers, technicians, and operators have insufficient opportunity to practice the complex skills involved in the operation of propulsion plants. For many of these skills, increased practice is so prohibitively expensive that Navy personnel have simply had to do without all but a minimal level of practice.

The need for propulsion simulation training is well documented. The Atlantic fleet, for example, has 663 fireroom-engineroom "watch sections" with approximately eight personnel in each section, and has an annual turnover of personnel of over 50 percent. This generates an annual training requirement of 332 sections per year just to remain even with crew turnover. Since 3 weeks of training on a full-scale 19E22 simulator are needed for each watch section, the 19E22 simulator can train only 17 watch sections annually. To meet the complete training requirement, the Atlantic fleet would need to have 20 simulators, each of which costs about 9 million dollars and requires a staff of 21 senior instructors. The fleet cannot afford this massive commitment.

It is now possible to implement inexpensive stand-alone computer-based training systems of sufficient power to provide training in the types of qualitative understanding that are needed for safe and effective propulsion plant operation. Five years ago, computers powerful enough to run a large-scale simulation cost several hundred thousand dollars and filled the better part of a large air-conditioned room. Today, such computers cost under a hundred thousand dollars and are about the size of a filing cabinet. All technology forecasts indicate that, in 5 more years, computers of this power will fit easily on a desktop and cost about ten thousand dollars. Besides providing more power in smaller packages, newer computer technologies have made it possible to (1) present animated diagrams in black-and-white and color, (2) provide voice output, and (3) accept input in graphics and spoken form.

Background

The STEAMER project is a 5-year research and development effort to implement and evaluate applications of this advanced computer technology to critical Navy training problems. The major goal of the project is to produce an inexpensive desktop-sized computer sytem for instruction in propulsion plant operation. By incorporating techniques developed in the fields of cognitive psychology and artificial intelligence, the training systems will, in addition to providing simulation, ease the instructor's load by incorporating an automated tutor capable of providing explanations and tutorial assistance to students as they progress through exercises in propulsion plant operation. Incorporating an automated tutor into an inexpensive, portable, simulation-based training system will greatly enhance the amount and quality of training available to Navy personnel.

To achieve these objectives, the STEAMER system combines several technological developments and areas of research. Besides the computer and numerical modeling technologies important for building a portable simulation system, STEAMER is incorporating developments in student modeling, knowledge representation systems, and qualitative reasoning. Student modeling techniques will make it possible for STEAMER to do more than simply correct student errors. Rather, by using errors to accumulate evidence about the student's underlying misconceptions, the system can provide tutorial assistance that enables the student to understand the nature of the error. Knowledge representation techniques make it possible for STEAMER to train the student about generic systems and procedures and show how they apply to the specific systems the student must operate.

Developments in the formalization of qualitative reasoning provide a set of modeling techniques to represent the conceptual relationships that form the basis of much of what an expert knows about the operation of complex systems. These techniques will make it possible for STEAMER to store and communicate to students the causal relations between changing events in the modeled propulsion plant.

The STEAMER project is designed to take advantage of the powerful computer hardware that is projected to continue its rapid decrease in size and cost. The STEAMER system includes a mathematical model of a propulsion plant system, a graphics interface to the mathematical model, and an intelligent automated tutor to teach students basic operating principles and procedures.

The combination of techniques used in STEAMER allows an approach that will produce a system providing both breadth and depth in training. Students will be able to oᵔerate a propulsion plant and "see" hundreds of its parameters. They will be guided through procedures, provided with intelligent advice and qualitative explanations about its workings, and allowed to experiment in depth with specific systems designed to teach underlying principles governing the operation of pumps, AC circuits, and behavior of gases and liquids.

STEAMER is a software system. It is important in developing the prototype of such a system to target it for hardware that will be available and inexpensive when the system is ready for production. It is equally important that such a system match the needs of potential users. These two goals are potentially conflicting. Developing a system for existing hardware ignores the enormous potential of soon-to-be-available computers. Developing a system for future hardware means that development must be done on large, expensive computers not readily accessible to users. This makes it difficult to get good feedback from intended users. To meet the first of these goals, the STEAMER system is being carefully developed to preserve portability onto hardware projected to cost about $10K to $20K in 1985. To meet the second goal, ensuring development in a way that meets the Navy's present and future training needs, an in situ development plan has been initiated. This involves providing at least one Navy school with a scaled-down version of the developing STEAMER system, training school personnel how to use the system, and working with school personnel to make video tapes of selected sequences to use as part of their curriculum. This provides a concrete basis for discussion, enables instructors and students to get a hands-on feel for STEAMER, and provides valuable feedback to the system developers. This plan is being coordinated with the Navy's Surface Warfare Officer School (SWOS) in Newport, Rhode Island, and with the Navy's Propulsion Engineering Steering Committee.

A significant part of the first-year effort has gone into building important tools and utility programs needed for further development. These include a graphics package, and graphics editor (Stead, 1981), a set of utilities for running, examining, and debugging large mathematical models (Roberts & Forbus, 1981), a constraint interpreter (Forbus, 1981), and techniques for qualitative simulation (Forbus & Stevens, 1981). The mathematics model utilities include methods for initializing a mathematical model, running and interrupting it, observing selected variables, and looking up variable names. Other utilities permit the restructuring of the model so that the combined states of a number of related variables can be summarized by a single variable. This is an important step in the move toward the integration of an intelligent tutorial capacity to the model. The graphics package and editor enable diagrams and other displays to be rapidly and easily constructed and modified. The ease of modification is important to allow incorporation of feedback and ideas from the intended users. At present, the STEAMER system consists of an engineroom propulsion plant mathematics model, an easy-to-use graphics interface, the graphics editor and graphics package for creating new diagrams, and parts of an intelligent tutoring component capable of explaining the operation of simple control devices and presenting instruction in relevant basic physics principles.

Purpose

The purpose of this report is to describe the current STEAMER prototype, including the mathematical model, graphics, explanation generation, procedures training and mini-labs, as well as the basic support software for further STEAMER development.

CURRENT STEAMER PROTOTYPE

User's Perspective

The Numerical Simulation and Graphics Interface

The STEAMER display consists of two adjacent television-sized screens--1 in color for diagrams and 1 in black and white for text. Figure 1 shows these two displays. The color screen is touch-sensitive. The student manipulates the simulated steam plant by touching displayed valves, causing them to open or shut, touching other components to turn them on or off, and adjusting other simulated components such as throttle valves. This style of use is so simple that it requires almost no instruction or practice to master. Changes in state of the components are indicated by color changes or changes on depicted indicators such as dials, thermometers, and digital readouts.

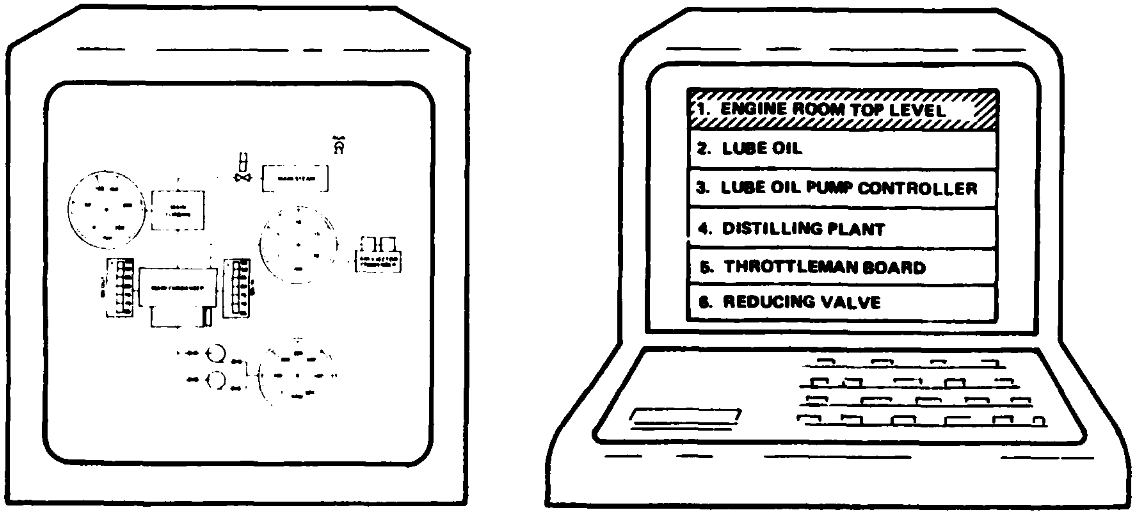


Figure 1. The two STEAMER display screens--color (left) for diagrams and black and white (right) for text.

Figure 2 shows a typical display. To change the throttle setting, the student simply touches the column above the throttle valve (to the left of "Main Steam"). As the throttle changes, turbine RPM and flow into the hotwell (below the main condenser) change accordingly. The hotwell indicator shows the resulting level changes by changing height. As the hotwell level changes, the flow through the condensate pumps (below the hotwell), governed by submergence level, changes accordingly and the flow rate is shown dynamically as blink rate on the pipes. Using this display, an instructor can illustrate such principles as submergence control by shutting off both pumps, allowing the hotwell level to rise, then turning on a pump, and showing the flow rate change as the hotwell level drops and stabilizes.

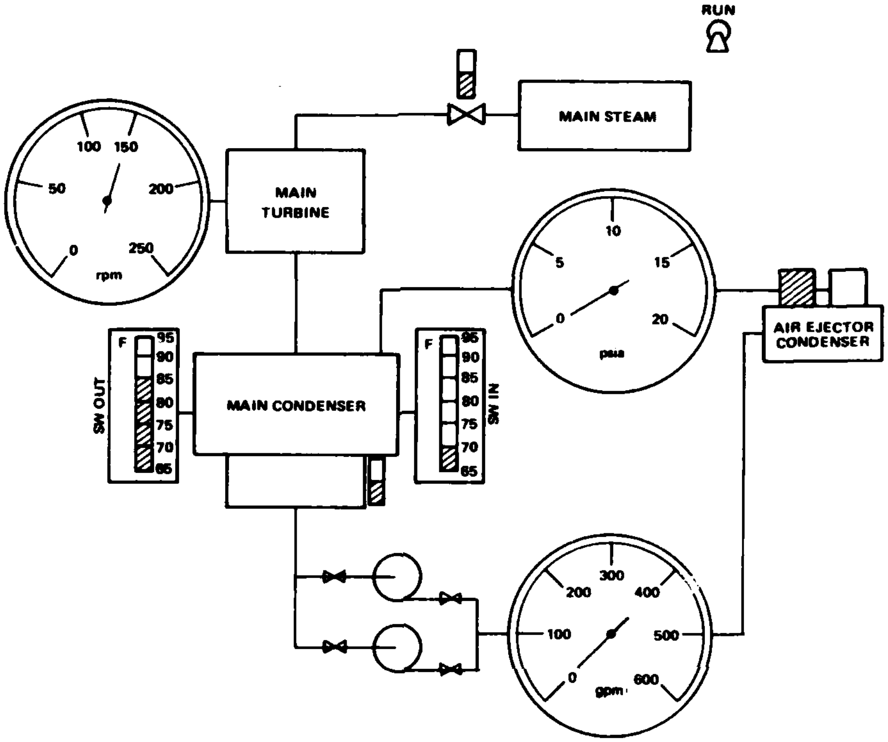


Figure 2. An interactive STEAMER display for the engineroom.

The schematic diagram available to the student in Figure 2 shows only 20 or so components and indicators. A steam plant is much more complicated than that. Using STEAMER, the student has access to other propulsion plant subsystems by selecting a particular view from a menu of choices displayed on the adjacent screen. For example, if the student selects "Main Engine Lube Oil" from the menu, the engineroom diagram is replaced with that shown in Figure 3. Using this diagram, the student or instructor can experiment with and observe the complex control dynamics in the lube oil system. He can close the throttle to decrease the shaft rpm (141 in Figure 3), and observe the oil pressure dropping, causing the pressure sensor (number 1) on the most remote bearing to close and the standby pump (LSOP 1 or 2) to come on line. He can shift the display to the lube oil pump controller to reconfigure the pumps and then go through the same exercise to see how each pump can serve either the role of standby or emergency.He can go to a casualty panel display and fail one of the lube oil pumps and then go through the exercise again to see how the emergency pump backs up the standby pump. He can observe the unloading valve opening and dumping oil to the sump if the shaft rotation increases and electrical pumps are not shut off. He can even observe the effects on pressure of such casualties as a clogged strainer.

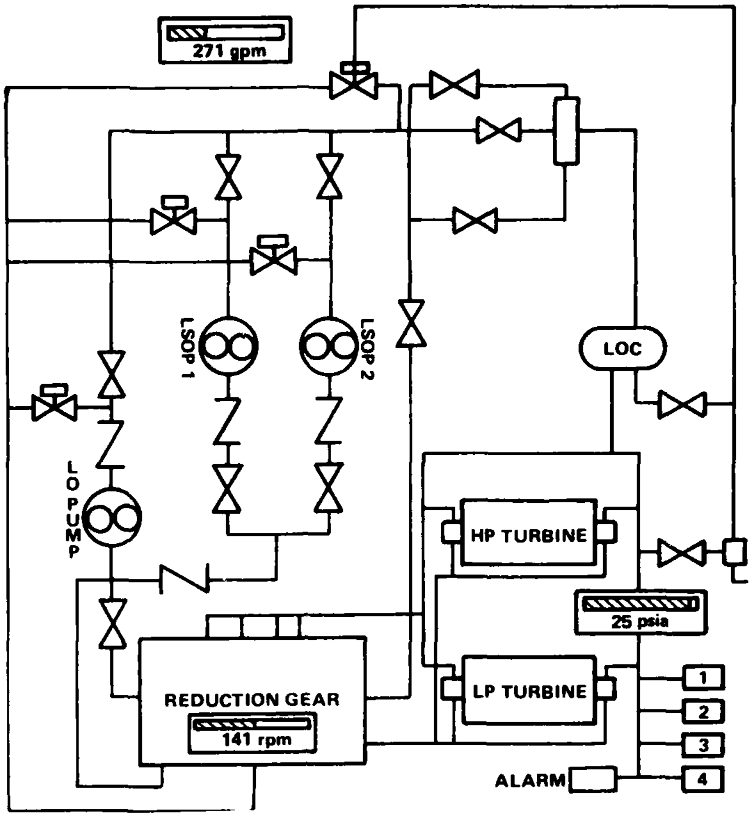


Figure 3. An interactive diagram of the main engine lube oil system.

The effectiveness of this interactive inspectable simulation is difficult to describe. On a ship, the main engine lube oil system is distributed across several places. Much of it is constructed from opaque material. Seeing it all and grasping what is happening is extremely difficult. This same problem applies to full-scale simulation mock-ups such as the Navy's 19E22 system. With a STEAMER display, the whole lube oil subsystem and its important parameters are available for inspection. The rapid control dynamics can be seen and understood in a glance. Because it is based on a simulation, the student or instructor can experiment with and observe the effects of various casualties without fear of damaging real equipment. Because it is based on general computer hardware, all of the major propulsion plant subsystems are available for use on a desktop or in a classroom.

Tutorial Component

The STEAMER system is designed to incorporate an intelligent tutorial component capable of providing students with guidance in plant operating procedures, instruction in basic operating principles, and explanation of component and subsystem operation. This component is the most difficult to design and implement; consequently, it is the least well-developed. At this time, the tutorial component consists of an explanation generation component capable of explaining the operation of feedback systems and a set of "minilabs" for teaching basic physics principles necessary to understand the plant. The integration of these pieces into the STEAMER system is not yet complete. They will be described independently.

Minilabs. Besides making available a whole propulsion plant, the STEAMER system allows smooth integration of instruction on basic principles with training on propulsion plant operation. For example, if a student does not understand the relationship between pressure and velocity head, a "minilab" using the same display hardware can be presented to illustrate those principles. Figure 4 shows a display from such a minilab, which consists of a constricted pipe with fluid flowing in it. In this version of the minilab, the student can take measurements of pressure and velocity and have them graphed in the corresponding position below.

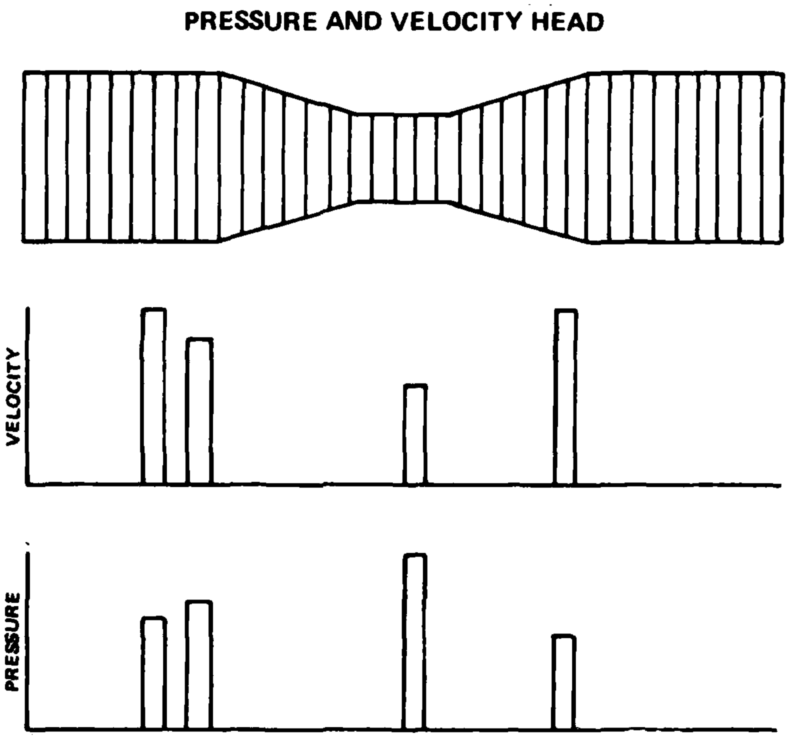


Figure 4. A display from a STEAMER minilab illustrating pressure and velocity head relationships.

Another experimental minilab, on pumps, allows a user to define a pump by drawing a graph on the CRT screen of its characteristic operation for different levels of submergence. Once drawn, STEAMER builds a runable model of the pump and allows the student to interact with it. He can raise the level of the hotwell, change the flow rate in, and change the head the pump is working against. He sees immediately the effects of these changes, watching the pump control the level in the hotwell by changing its capacity, and watching the pump head change the pump capacity and the hotwell level. He can watch the flow change, the hotwell level change, and the pump capacity adjust until the system stabilizes. This level of interaction allows the student to see directly the interrelationships of the relevant system parameters such as tank level, pump capacity, and system head. It allows (1) the student to see how these parameters correspond to the graphs typically used to communicate this information in technical manuals, and (2) a level of interaction that encourages the student to ask questions by experimenting to see what the answers are.

This integration of basic principles and plant operation is even closer than so far described. In addition to the experimental minilab, STEAMER incorporates a display that allows a student to control the condensate system in the numerical simulation and observe parameters of it on a graph. The graph plots the capacity vs. submergence as the steam plant runs. The student or instructor can quite literally see the curves governing pump operation in the working system.

One of the premises of the STEAMER effort is that knowledge of these basic principles is important. This conclusion comes from extensive interviews of Navy training personnel at the SWOS and Great Lakes schools. STEAMER provides, for the first time, an instructional medium in which training on basic principles can be easily integrated with training on operational procedures. This integration appears to be a major advantage over traditional instructional media. By plotting graphs dynamically at the same time the system is running, the problems associated with reading graphs are alleviated--students are able to understand the graph easily. By showing the principles in the context of an operating plant, their relevance is apparent and the meaning of the principles in terms of how and why the plant operates as it does is clear.

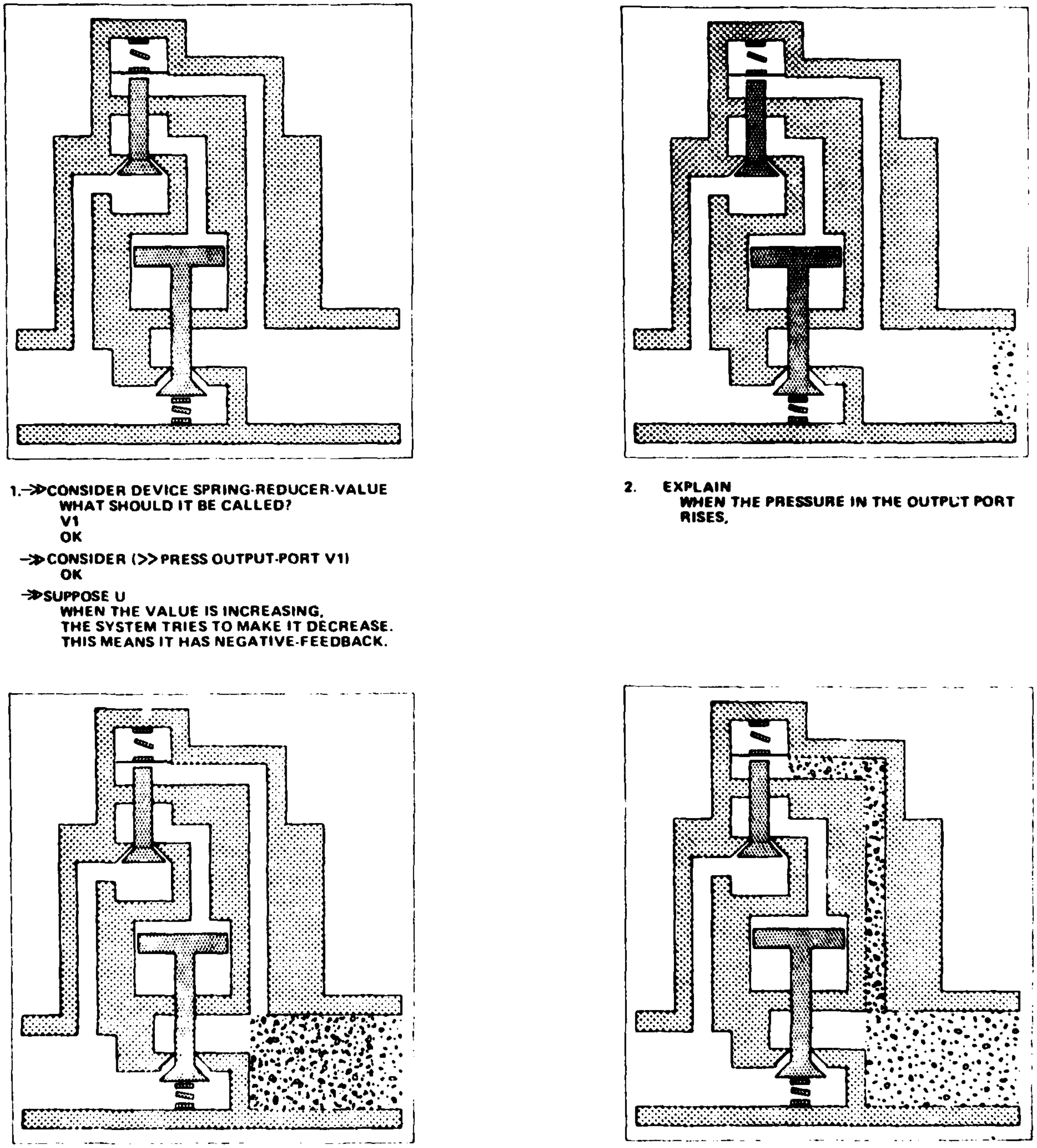
Explanation generation. An important goal of STEAMER is to provide students with understandable explanations. Generating explanations requires that the instructional system must itself have some understanding of the topic, preferably close to the kind the student should have. There is growing evidence that human understanding of physical systems is based on qualitative models of those systems. This evidence comes from psychological studies (Larkin, McDermott, Simon, & Simon, 1980; Stevens, Collins, & Goldin, 1979) and is supported by successes in artificial intelligence in actually constructing systems that reason about physical situations using qualitative models (deKleer, 1979; Forbus, 1980).

Consider the following explanation of an air-operated pilot valve:

> As the controlled pressure (discharge pressure from the diaphragm control valve) increases, increased pressure would be applied to the diaphragm of the direct acting control pilot. The valve stem would be pushed down and the valve in the control pilot would be opened, thus sending an increased amount of operating air pressure from the control pilot to the top of the diaphragm control valve. The increased operating air pressure acting on the diaphragm of the valve would push the stem down and--since this is an upward seating valve --this action would open the diaphragm control valve still wider. (Bureau of Naval Personnel, 1970, p. 383.)

This explanation is comprised of a set of events, each describing a qualitative change in some part of the device. The explanation is linearized and describes how physical effect is passed from one component to another. It ignores the true temporal changes where those things that are happening are happening continuously and simultaneously. (Other reports describe the form of explanations in more detail (Forbus & Stevens, 1981; Stevens & Steinberg, 1981).)

Figure 5 presents selected frames from an explanation generated by STEAMER. In Frame 2, the student asks for an explanation. Each panel of the explanation is drawn from the actual computer display that a student sees. Successive panels denote successive states of the display. The device described is a spring-loaded reducing valve, a common type of control device that serves to supply steam at a constant reduced pressure to a set of varying loads. Demonstrations of this tutorial component have been given to



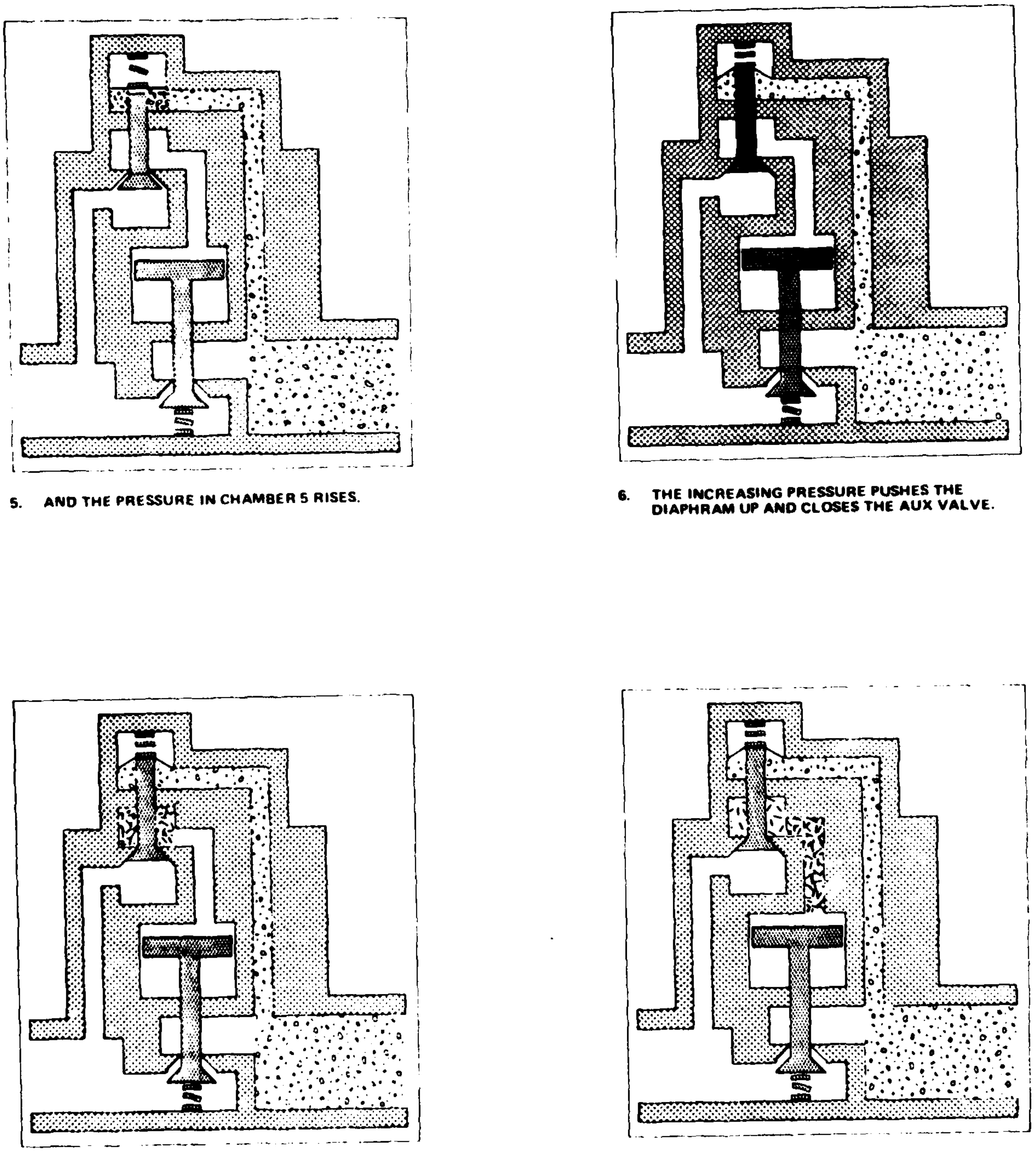
```
1.->>CONSIDER DEVICE SPRING-REDUCER-VALUE
      WHAT SHOULD IT BE CALLED?
      V1
      OK
 ->>CONSIDER (>>PRESS OUTPUT-PORT V1)
      OK
 ->>SUPPOSE U
      WHEN THE VALUE IS INCREASING,
      THE SYSTEM TRIES TO MAKE IT DECREASE.
      THIS MEANS IT HAS NEGATIVE-FEEDBACK.
```

```
2.    EXPLAIN
        WHEN THE PRESSURE IN THE OUTPUT PORT
        RISES,
```

3 THE PRESSURE IN CHAMBER D RISES,

4. THE PRESSURE IN THE LOW PRESSURE PORT RISES,

Figure 5. Successive frames of the explanation generated for a spring-loaded reducing valve.



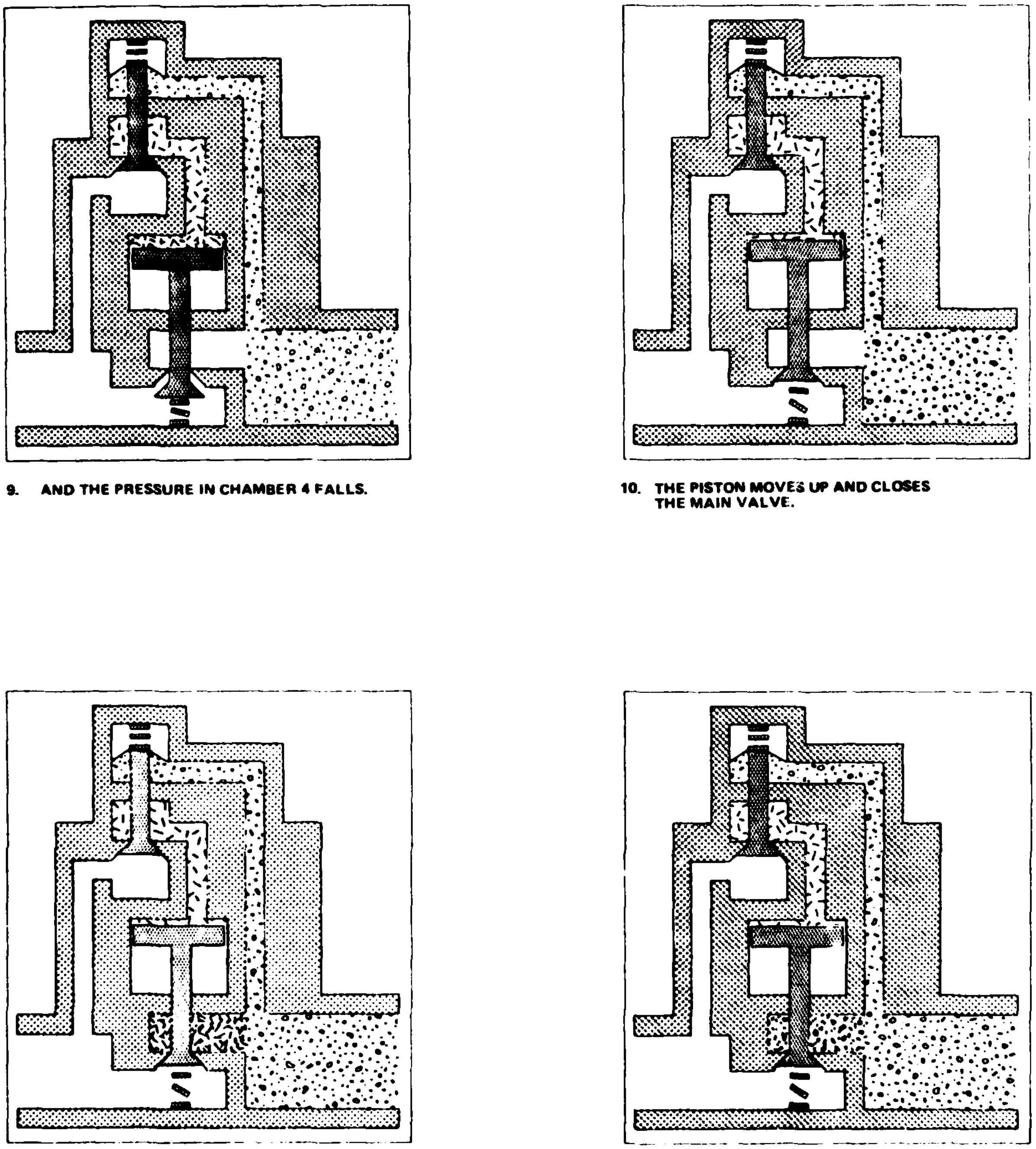
5. AND THE PRESSURE IN CHAMBER 5 RISES.

6. THE INCREASING PRESSURE PUSHES THE DIAPHRAM UP AND CLOSES THE AUX VALVE.

7. THE PRESSURE IN THE AUX VALVE'S OUTPUT SIDE FALLS.

8. THE PRESSURE IN THE PISTON STEAM PORT FALLS.

Figure 5. (Continued).



9. AND THE PRESSURE IN CHAMBER 4 FALLS.

10. THE PISTON MOVES UP AND CLOSES THE MAIN VALVE.

11. THE PRESSURE IN THE MAIN VALVE'S OUTPUT SIDE FALLS.

12. ALSO WHEN THE PRESSURE IN CHAMBER 0 RISES THE PRESSURE IN THE MAIN VALVE'S OUTPUT SIDE RISES.
NOTICE THAT WHEN THE PRESSURE IN THE OUTPUT PORT RISES IT MEANS THE PRESSURE IN THE MAIN VALVE'S OUTPUT SIDE RISES. HOWEVER, IT ALSO CAUSES THE SYSTEM TO ACT SO THAT THE PRESSURE IN THE MAIN VALVE'S OUTPUT SIDE FALLS. THIS MEANS IT EXHIBITS NEGATIVE FEEDBACK.

Figure 5. (Continued).

SWOS representatives and to members of the Navy's Propulsion Engineering Steering Committee. They have been uniformly positive about its utility.

The particular goal of this tutorial component is to explain feedback systems. STEAMER is capable of recognizing and explaining instances of negative feedback, positive feedback, and stable, unstable and open-loop systems. The explanation techniques used make possible a learning environment in which students can experiment with complex devices and see explanations of the effects of various changes.

STEAMER Design

The design of STEAMER emphasizes flexibility. New displays can easily be constructed and connected to the mathematical model. Once constructed, they can be easily modified. This is important, both because STEAMER is to be used for teaching about propulsion plants, which change as new technologies become available, and because instructors' ideas and curricula change. The STEAMER design allows instructors and curriculum planners to modify it to fit their course material rather than forcing the course material to be modified to fit STEAMER.

The STEAMER system contains a large amount of software. It has been the case with many software projects that, as the software systems grow in size, they become fragile, and difficult to modify and maintain. This occurs because, as the systems grow, they become difficult for programmers to understand. The most successful solution to this problem is to design the system as a set of independent, manageable modules that communicate in well-specified ways. The STEAMER design strongly adheres to this methodology, is highly modular, and has proven easy for programmers to work with. The basic system consists of three major parts: (1) a numerical simulation model of a 1078-class propulsion plant, (2) a set of diagrams depicting different subsystems of the propulsion plant, and (3) a set of software objects, called taps, that connect the diagrams to the simulation so that it can be controlled and inspected. In addition to this inspectable, controllable simulation, STEAMER includes a graphics editor for creating new diagrams and a tutorial component.

STEAMER Numerical Simulation

The STEAMER numerical simulation is directly derived from the Navy's 19E22 propulsion plant trainer. It is a straightforward simulation, modelling a 1078-class plant down to the level of components (e.g. valves, pumps, motors, and switches) and thermodynamic variables (e.g., temperatures, pressures, and flow rates). The simulation and facilities for debugging it are described in detail in Roberts and Forbus (1981).

Graphics Interface

A displayed view of the steam plant is constructed out of graphical building blocks, some of which represent common geometric shapes, and some, the Navy symbology used in steam plant drawings. A view in the STEAMER system is not painstakingly laid out point by point and line by line as it would be in a typical graphics system, but is built by arranging these component graphical objects on the screen.

Icons. The internal representation of a graphical object is called an icon. Icons can be thought of as prototype objects from which displays can be constructed. Figure 6 shows many of the icons currently available to construct diagrams. Having such an "icon library" solves one major problem normally associated with building graphics displays: It greatly decreases the labor necessary to create and modify a new display. STEAMER

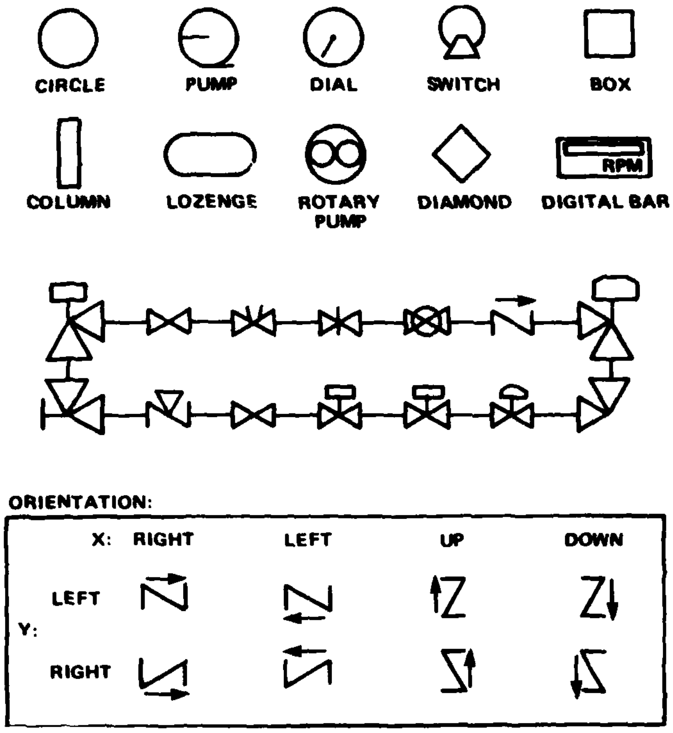


Figure 6. A sampler of STEAMER graphical objects.

icons also solve a second problem. Making a graphics display change dynamically to reflect the state of a mathematical model is normally a labor-intensive task. STEAMER icons are active objects that embody procedures for changing the display state (e.g., pump color, dial reading, numerical value) when a mathematical model variable changes. By providing prototype objects that embody procedures for changing the display state, STEAMER icons make creating new displays an easy task. A discussion of the types of icons available, their internal structure, and methods for creating them is included in the appendix.

Connecting Icons to the Simulation: Taps

The STEAMER icons embody a great deal of knowledge about drawing, and some information that enables them to react to user input. This section describes the method of connecting icons to the STEAMER numerical simulation.

A class of objects called taps serves as the connection between the icons and the variables manipulated by the numerical simulation. Figure 7 shows the organization. Taps are the sole interface between icons and the numerical simulation. When the variables in the simulation change, the icons on the screen must be sent an updated value to show. Conversely, an icon, when touched, returns a potentially showable value that must be converted into some change in the controlling variables in the simulation. Two of the messages understood by taps, PROBE and SET, correspond to these two cases.

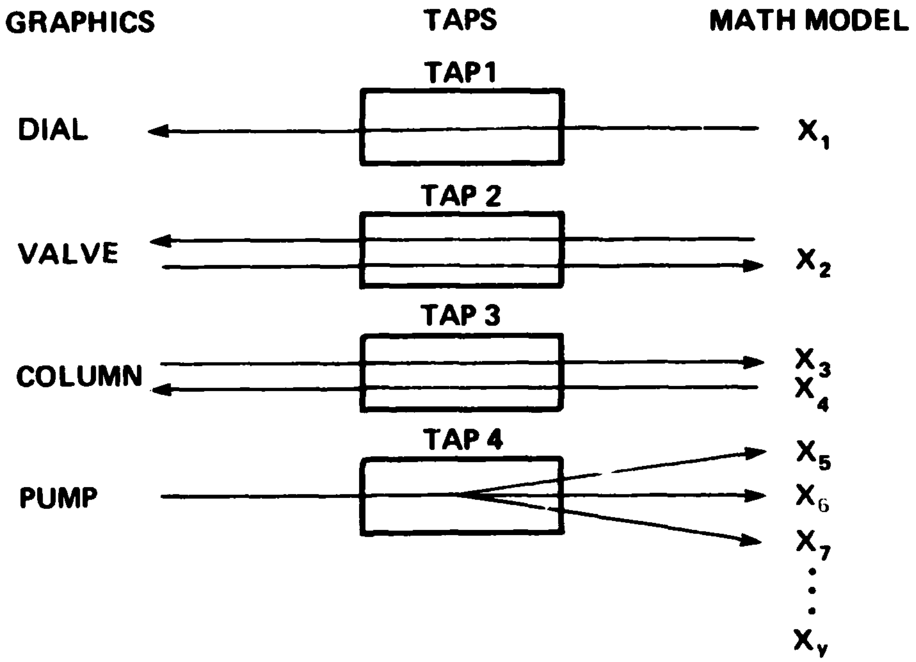


Figure 7. The relationship between taps, the math model and the Icons.

Tutorial Component

The two parts of the tutorial component that currently exist are the minilabs and the explanation generation component. The minilabs are implemented in a straightforward way, making use of the STEAMER icon library, graphics editor, and taps. The ease with which this has been accomplished is another example of the easily-used modular design.

The qualitative explanations require additional techniques. In particular, generating an explanation is not straightforward because numerical simulations provide only a set of continuously changing parameters. Explanations are much more discrete and qualitative. Consequently, STEAMER is using a technique called qualitative simulation.

The basic idea for a qualitative simulation comes from the observation that, when trying to understand or explain a device, people often use a description of how parts of it change when some influence is applied to the system. The changes in physical quantities such as pressure or the position of a valve are typically described by indicating the direction of the changes. Thus, for a pressure, the changes are "up," "down," or "constant."

The sequence of events in such a simulation depends on how the components of the device are connected together; changes in one quantity can affect only those other quantities related to it through some sort of connection. This means that complex devices can be modelled by specifying how a set of component models is connected together. Once certain assumptions about the operation of the device are made, the effects of a change on one part can be found by local propagation through the component models of the device. This is the essence of the incremental qualitative (IQ) analysis formalized by deKleer (1979) for electronic circuits.

The component models used so far in STEAMER are very simple. Spaces in a device are modelled by chambers, with ports and pipes transmitting pressure changes through them. Valves are modelled in terms of changes in their openings. When the valve opening increases, the pressure in the input side decreases and the pressure in the output side increases. When the valve opening decreases, the opposite happens. A translator models collections of components that turn the change in one type of quantity into another (such as the diaphragm/spring/valve stem combination that causes a change in pressure to change the position of a valve).

The descriptions are expressed in the constraint language CONLAN, described in an earlier report (Forbus, 1981). A qualitative simulation of a device is obtained by simply specifying a value from the IQ algebra for a selected part of the device (such as the output port for the spring reducer valve) and running the constraint interpreter on it. In this system, the parameter is interpreted as the controlled parameter of the device. The interpreter deduces values for as many of the component quantities as it can by running the rules associated with the component models. It records the results of this qualitative simulation as a graph structure. The nodes of this structure represent the quantities and the arcs represent the rules used to deduce them. This description of the history of the simulation is used as the basis for generating an explanation.

While the structure of the qualitative simulation is similar to those normally used by people, its internal representation is not easy to understand. By translating it into English and using graphical cues, it is turned into a coherent explanation. This is accomplished by a simple grammar and template scheme that transforms the computation paths in the constraint network into an interleaved English and graphical presentation.

Results of analyzing the simulation are handled in the same fashion. A stored template provides an English explanation of the results, filled in with the phrases that describe the particular events in the device under consideration that led to the conclusions.

Current State of STEAMER

The STEAMER interface currently consists of a set of about 25 displays. That number is growing rapidly. Using the STEAMER graphics editor, creating a new display takes about 2 to 8 hours, depending on its complexity. Representatives of the Navy's SWOS school in Newport participated in a day-long review of the current STEAMER system and many of their suggested modifications have been incorporated.

All of the original expectations for STEAMER still seem justified. Even though the demonstration system has just become usable, its potential as a highly beneficial training aid is obvious. In informal experiments, people who know nothing about propulsion plants were shown the main engine lube oil system. They were able to run it with only a few minutes of instruction. More importantly, they were able to run, observe, and understand the operation of the simulated lube oil system itself. On January 27, 1981, a demonstration of the system was provided for evaluation by representatives of the Navy's SWOS school. The evaluation was positive with respect to the current STEAMER system and with respect to planned developments.

CONCLUSIONS

It may be concluded from the demonstration system that the motivations for development of STEAMER were correct:

1. The system is easy to use and consequently will allow greatly increased practice on complex operational skills.

2. The system will increase the quality of instruction by introducing the capability of allowing students to inspect and operate a propulsion plant at various conceptual levels, and by providing for automatically generated explanations and tutorials.

3. Projected hardware trends have proven correct and it will be possible in the next phase of the project to move the STEAMER system to an inexpensive portable computer.

4. The software design of STEAMER allows flexibility to programmers and curriculum developers in adding and modifying STEAMER capabilities.

5. The prototypes of various parts of STEAMER have been well received by prospective users within the Navy training community.

FUTURE DIRECTION

Because of care in design and attention to compatibility issues, STEAMER software is transportable across a number of different computers. To facilitate early development, STEAMER was implemented on a large time-sharing system. The next phase of the project will add the numerical simulation and diagrams to bring STEAMER up to a complete propulsion plant model and move the software to a smaller, more portable computer. The project is continuing its coordination with the Navy training community by arranging to place microprocessor-based systems at SWOS for preliminary tryout by students. This is expected to occur early in 1982. By the end of 1982, the prototype of the STEAMER simulation and graphics interface will be available in a form suitable for use by Navy training personnel. By the end of 1984, the system will be available on desktop-sized computers and incorporate not only the simulation and interface, but also the intelligent tutorial component.

REFERENCES

Bureau of Naval Personnel. Principles of naval engineering. U.S. Government Printing Office, 1970.

deKleer, J. Causal and teleological reasoning in circuit recognition (Ph.D. Thesis). Massachusetts Institute of Technology, September 1979.

Forbus, K. A study of qualitative geometric knowledge in reasoning about motion (Master's Thesis). Massachusetts Institute of Technology, February 1980.

Forbus, K. D. Project STEAMER: IV. A primer on CONLAN--a constraint-based language for describing the operation of complex physical devices (NPRDC Tech. Note 81-26. San Diego: Navy Personnel Research and Development Center, August 1981.

Forbus, K. D., & Stevens, A. Project STEAMER: III. Using qualitative simulation to generate explanations of how to operate complex physical devices (NPRDC Tech. Note 81-25). San Diego: Navy Personnel Research and Development Center, August 1981.

Goldberg, A., & Kay, A. Smalltalk-72 instructional manual (Tech. Rep. SSL-76-6). Palo Alto, CA: Xerox Palo Alto Research Center, 1976.

Larkin, J., McDermott, J., Simon, D. P., & Simon, H. A. Expert and novice performance in solving physics problems. Science, 1980, 208, 1335-1342.

Roberts, B., & Forbus, K. Project STEAMER. V. Mathematical simulation of STEAMER propulsion plant (NPRDC Tech. Note 81-27). San Diego: Navy Personnel Research and Development Center, August 1981.

Stead, L. Project STEAMER: II. User's manual for the STEAMER interactive graphics package (NPRDC Tech. Note 81-22). San Diego: Navy Personnel Research and Development Center, August 1981.

Stevens, A., & Steinberg, C. Project STEAMER: I. Taxonomy for generating explanations of how to operate complex physical devices (NPRDC Tech. Note 81-21). San Diego: Navy Personnel Research and Development Center, August 1981.

Stevens, A., Collins, A., & Goldin, S. Misconceptions in students' understanding. International Journal of Man-Machine Studies, 1979, 11, 145-156.

APPENDIX

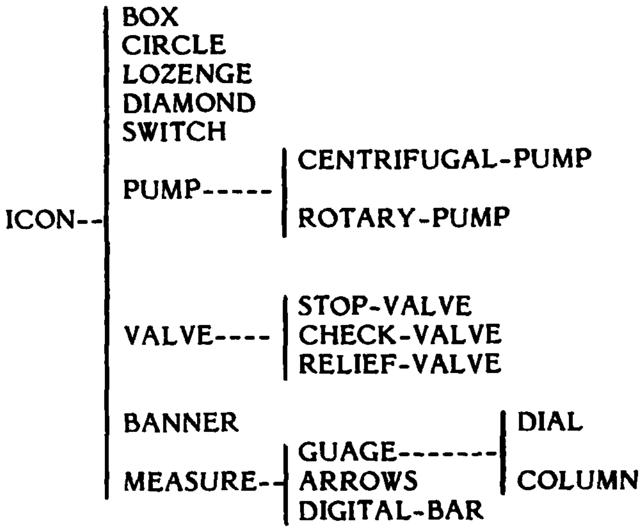
DESCRIPTION OF GRAPHICS INTERFACE

DESCRIPTION OF GRAPHICS INTERFACE

Icons

Hierarchy of Icon Classes

Icons are organized into a hierarchy of classes to maximize sharing among graphics objects that have similar characteristics. The hierarchy is shown below.



Icon is the most general object class. It defines the functionality common to all icons, but has no shape. Four subclasses of ICON--BOX, CIRCLE, DIAMOND, and LOZENGE (straight sides, rounded ends)--have noncommital shapes. Three other subclasses of ICON--PUMP, VALVE, and SWITCH--represent common components of a steam plant in their conventional schematic form; PUMP and VALVE are further differentiated into subclasses. These icons all can show discrete states of the device they represent.

The remaining icons are designed to show continuously varying values. A BANNER displays an arbitrary string of text. The subclasses of the MEASURE icon--ARROWS, DIGITAL-BAR, and GUAGE (incorporating DIAL and COLUMN)--display real-valued numerical information. ARROWS icons are small arrowheads whose blink rate can be varied; they are used to indicate flow in pipes. DIAL icons serve as pressure gauges and COLUMN icons, as thermometers. DIGITAL-BAR icons combine a digital readout with a column to show directly change and relative magnitude within a prescribed range.

The internal structure of an ICON. Icons possess a good deal of internal information besides the basic geometric parameters needed to draw them. All icons have the ability to alter dynamically some characteristic of their appearance to indicate a changing property of the system they are depicting. Icons have a local coordinate system, a basic outline color, a label, and a set of possible states (the defaults are ON and OFF). Different states are typically associated with unique colors (the defaults are GREEN and RED, respectively). Blinking colors are just another kind of available color, so blinking icons are subsumed by this general notion of color.

The STEAMER icon library is a class structure in the sense used in the Smalltalk language (Goldberg & Kay, 1976). Each icon class defines a set of named variables and procedures. Classes are arranged in a hierarchy and both variables and procedures are inherited by subclasses. Each subclass can add new variables and procedures or modify the meaning of those appearing above it in the hierarchy. The class definition describes a generic object of which any number of instances can exist. Creating a display creates a set of instances of objects from the library. Every instance has its own local variables and access to all the procedures defined for (or inherited by) its class.

Each icon class also has procedures associated with it that collectively define the behavior of its instances. Every icon must be able at least to draw and erase itself from the screen, modify its appearance to indicate its current state, determine whether a point on the screen lies within it, and highlight itself. The name of an action serves as a key to select among all the procedures defined for an icon. Actions may involve manipulating the icon in some fashion or returning some information about the icon (e.g., the coordinates of its center). The meaning of a procedure name is constant across all icons, although the implementation of it may differ from icon to icon. The hierarchical organization of the icon classes eliminates the need to duplicate the definition of an action for each new subclass. Procedures shared by all icons need only be defined once for the ICON class. They will then be inherited by all other icons. New procedures can be introduced in any subclass and thereby become available to all of its subclasses further down in the hierarchy.

Icons can differ sufficiently so that new procedures have to be defined to perform the same actions in different icons but, since the associated procedure names are held in common, a programmer is freed from knowing the peculiarities of each one. This is another example of the modular programming that keeps STEAMER software easily manageable.

Communication with icons is carried on within a message-passing paradigm. As implemented, icons are autonomous objects having a well-defined recognizable data type and possessing some amount of local storage. "Messages" in the form of procedure names are "sent" to objects to invoke some action. Icons in general handle such messages as DRAW, REDRAW, SHOW, ERASE, FILL, DRAW-LABEL, and ERASE-LABEL.

The graphics languag on which icons are built supports coordinate system transformation: mapping of coordinates to provide scaling and translation, reflection, and rotation. Icons are assumed to be drawn inside a rectangular region on the screen whose size is determined by the values specified in a COORDINATES message. The details of the basic graphics software are described in Stead (1981).

Creating an Icon: The Graphics Editor

Creating a new STEAMER diagram is done using the STEAMER graphics editor. Using it, a new diagram can be created rapidly and easily or old diagrams can be modified.

The icons to be put in a diagram are first created by developing instances of icon classes and supplying these new instances with the information relevant to icons of its type. Most characteristics have default values to minimize the amount of information that must be given to draw common cases. The following is typical of the computer code to define a valve:

```
(setq V23 (instance valve-icon-class
                construction      'globe
                x-orientation     'down
                label             "Gland Seal Steam"
                labelposition     'right
                labelfont         'small))
```

This expression will create an instance of the valve icon class. It will be a globe type valve oriented downward on the screen display. The label "Gland Seal Steam" will appear in small print font just to the right of the valve. The only truly necessary piece of information lacking in this specification is the screen location coordinates for valve V23. One could explicitly install them by typing them in, but the graphics editor provides an easier way. If the designer simply touches the position on the screen where V23 should go, the graphics editor will supply the appropriate coordinates.

The editor works by manipulating instances of icons. When given a command to add a new graphics object to the developing diagram, it prompts the user to indicate the object location. The location is indicated to the editor by touching the screen--usually twice so the size as well as location can be indicated. The editor then displays the object at that location. The graphics editor allows objects in the diagrams to be changed--it has commands to "redo" and "undo" objects in the diagrams. It can search the diagram, highlighting each object in turn until the one to be modified is reached. To provide draftsman-quality diagrams, the editor incorporates a feature for (1) aligning diagram parts along horizontal, vertical, and diagonal axes and (2) providing uniform scaling of objects in different parts of the diagram. The most unique feature of the editor is that it actually creates not a screen image but, rather, a computer program to draw the diagram. This does not only solve some potential storage problems, but also allows the diagrams to be easily incorporated into the STEAMER system.

Taps

Taps are objects in the same sense that icons are. They store three important pieces of information: the name of an icon manipulated by the tap, a procedure executed when the tap is probed (its value is sent to the icon in a SHOW message), and a procedure evaluated when the tap is set. Taps are sent messages such as READ, PROBE, SET, CLAIM, and INIT to pass information from the simulation to the display and vice versa.

Subclasses of the general TAP class provide mapping between simulation variables and the icons. TAP subclasses are shown below:

```
        |DISCRETE-TAP---- |BINARY-TAP
TAP---- |
        |CONTINUOUS-TAP
```

Variables in the model that represent numerical values such as temperatures or pressures are linked to the graphics icons, which depict them via instances of the CONTINUOUS-TAP class. Variables that represent discrete, "all or nothing" values, such as the availability of electrical power in a circuit, are linked to icons via instances of the DISCRETE-TAP class. The tap class can make the mapping more convenient for the user.

For example, while the state of valves in the simulation is represented by the two values T and NIL, it is more meaningful to the user to label the valve icon states OFF and ON. The BINARY-TAP class performs this mapping automatically, letting one SET a valve "ON," for instance.

DISTRIBUTION LIST

Director of Manpower Analysis (ODASN(M))
Chief of Naval Operations (OP-01), (OP-11), (OP-12) (2), (OP-115) (2), (OP-140F2), (OP-987H)
Chief of Naval Material (NMAT 00), (NMAT 08L)
Chief of Naval Research (Code 200), (Code 440) (3), (Code 442), (Code 448)
Chief of Information (OI-213)
Chief oi Naval Education and Training (02), (N-5), (N-9)
Chief of Naval Technical Training (016)
Commander in Chief U.S. Atlantic Fleet
Commander in Chief U.S. Pacific Fleet
Commander Fleet Training Group, Pearl Harbor
Commander Naval Air Force, U.S. Atlantic Fleet
Commander Naval Air Force, U.S. Pacific Fleet
Commander Naval Sea Systems Command
Commander Naval Surface Force, U.S. Atlantic Fleet
Commander Naval Surface Force, U.S. Pacific Fleet
Commander Naval Military Personnel Command (NMPC-013C)
Commander Training Command, U.S. Atlantic Fleet
Commander Training Command, U.S. Pacific Fleet
Commanding Officer, Naval Regional Medical Center, Portsmouth, VA
Commanding Officer, Naval Training Equipment Center (Technical Library)
Commanding Officer, Surface Warfare Officers School Command
Director, Naval Civilian Personnel Command
Director, Naval Education and Training Program Development Center Detachment, Memphis
Director, Training Analysis and Evaluation Group (TAEG)
Superintendent, Naval Postgraduate School
Commander, Army Research Institute for the Behavioral and Social Sciences, Alexandria (PERI-ASL)
Chief, Army Research Institute Field Unit, Fort Harrison
Commander, Air Force Human Resources Laboratory, Brooks Air Force Base (Scientific and Technical Information Office)
Commander, Air Force Human Resources Laboratory, Lowry Air Force Base (Technical Training Branch)
Commander, Air Force Human Resources Laboratory, Williams Air Force Base (AFHRL/OT)
Commander, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base (AFHRL/LR)
Superintendent, U.S. Coast Guard Academy
Defense Technical Information Center (DDA) (12)